**AI Assisted Coding - Lab Test 2**

**Htno : 2503A52L14**

**Subgroup B**

**B.1 — [S15B1] Apply surge/penalty rules (conditionals)**

**Prompt :**

"Write a Python function to calculate ride fares based on time and distance, applying a surge multiplier strictly after 18:00. Include rounding to 2 decimal places and a quick test."

**Code :**

def calculate\_fares(rides, base\_per\_km, surge\_multiplier):

    """

    Calculate fares for a list of rides based on time and distance.

    Args:

        rides (list): List of dictionaries with 'time' (HH:MM) and 'km' (float).

        base\_per\_km (float): Base rate per kilometer.

        surge\_multiplier (float): Multiplier for surge pricing after 18:00.

    Returns:

        list: List of fares rounded to 2 decimal places.

    """

    fares = []

    for ride in rides:

        time = ride['time']

        km = ride['km']

        # Parse the hour and minute from the time string

        hour, minute = map(int, time.split(':'))

        # Determine if surge pricing applies

        if hour > 18 or (hour == 18 and minute > 0):

            fare = km \* base\_per\_km \* surge\_multiplier

        else:

            fare = km \* base\_per\_km

        # Round the fare to 2 decimal places

        fares.append(round(fare, 2))

    return fares

# Quick test

if \_\_name\_\_ == "\_\_main\_\_":

    rides = [{'time': '08:00', 'km': 3.0}, {'time': '18:30', 'km': 5.0}]

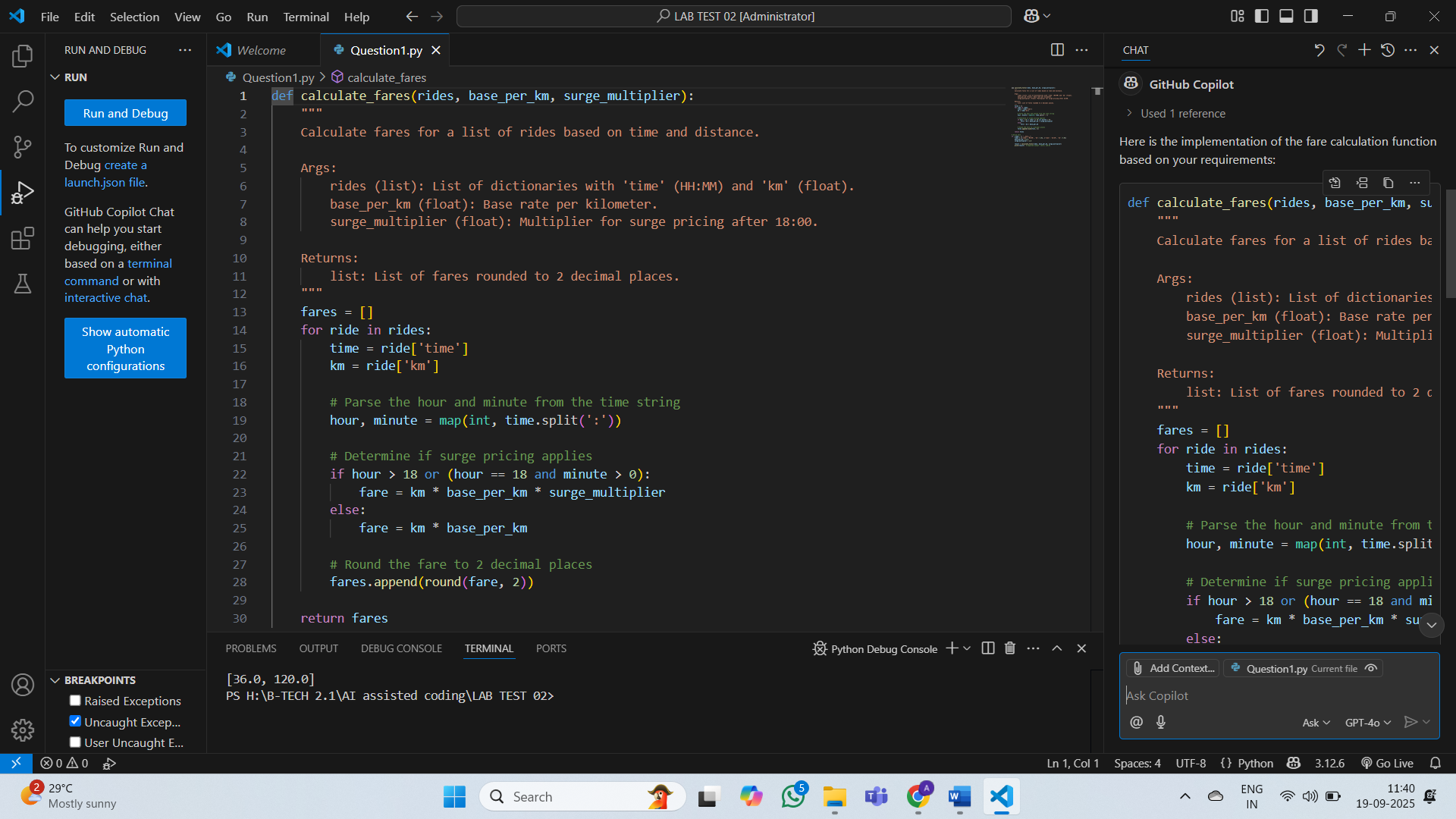
    base\_per\_km = 12.0

    surge\_multiplier = 2.0

    result = calculate\_fares(rides, base\_per\_km, surge\_multiplier)

    print(result)  # Expected output: [36.0, 120.0]

**Output :**



**Explanation :**

This code calculates ride fares based on distance and time, applying a surge multiplier for rides starting after 18:00. It parses the time, checks the surge condition, computes the fare, and rounds it to 2 decimal places. A quick test is included to verify the function with sample input.

**B.2 — [S15B2] Debug rolling mean (off-by-one)**

**Prompt :**

"Write a Python function to calculate the rolling mean of a list with a given window size, ensuring all valid windows are included. Add error handling for invalid window sizes and include a quick test."

**Code :**

def rolling\_mean(xs, w):

    """

    Calculate the rolling mean of a list of numbers with a given window size.

    Args:

        xs (list): List of numbers.

        w (int): Window size.

    Returns:

        list: List of rolling means.

    """

    if w <= 0 or w > len(xs):

        raise ValueError("Window size must be greater than 0 and less than or equal to the length of the list.")

    means = []

    for i in range(len(xs) - w + 1):  # Fix: Include all valid windows

        window = xs[i:i + w]

        means.append(sum(window) / w)

    return means

# Quick test

if \_\_name\_\_ == "\_\_main\_\_":

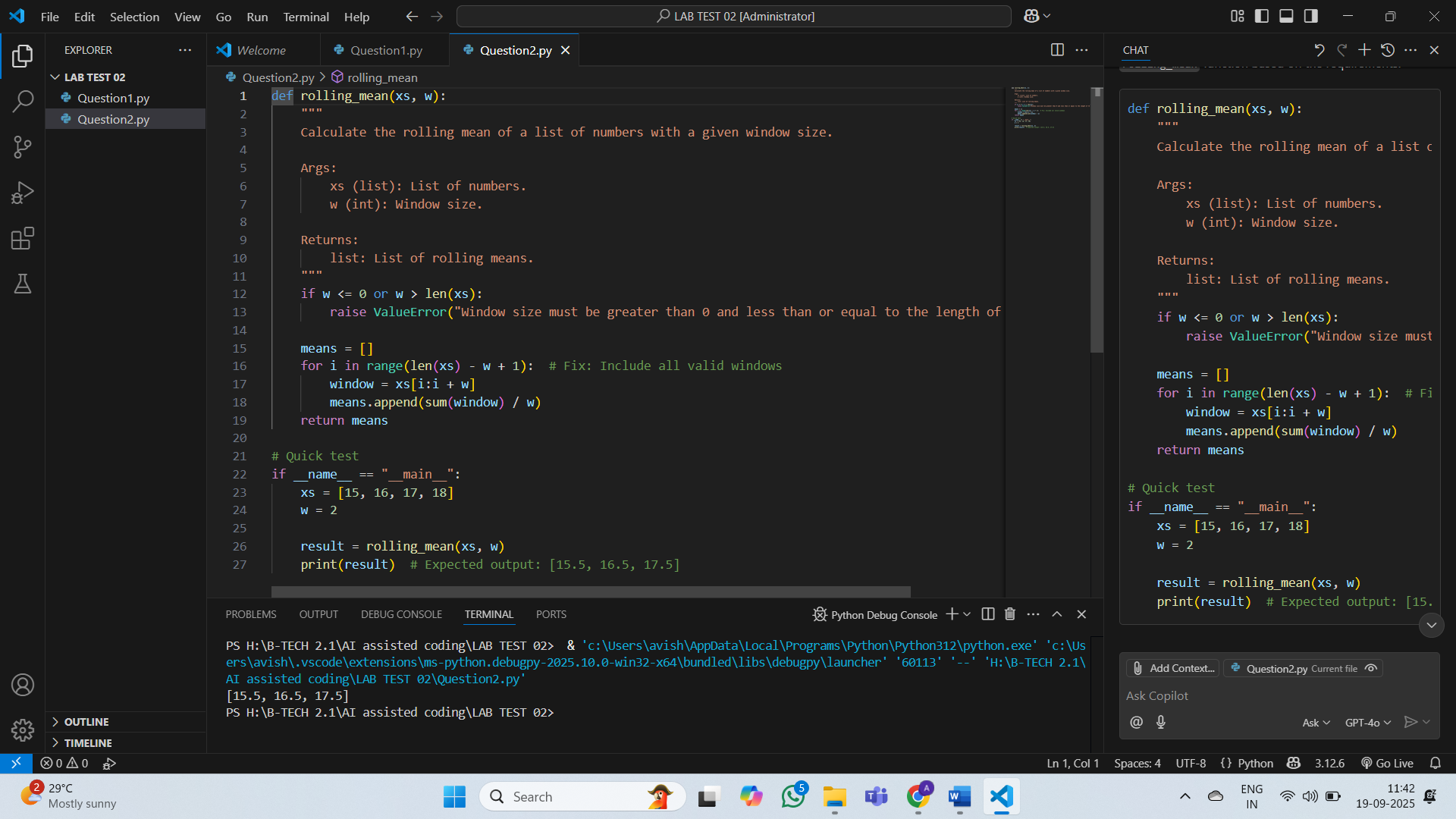
    xs = [15, 16, 17, 18]

    w = 2

    result = rolling\_mean(xs, w)

    print(result)  # Expected output: [15.5, 16.5, 17.5]

**Output :**



**Explanation :**

1. The [rolling\_mean](vscode-file://vscode-app/c:/Users/avish/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") function calculates the rolling mean of a list of numbers for a given window size by iterating through all valid windows ([len(xs) - w + 1](vscode-file://vscode-app/c:/Users/avish/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ")) and averaging the values in each window. It raises a [ValueError](vscode-file://vscode-app/c:/Users/avish/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") for invalid window sizes (<= 0 or > length of the list).
2. A quick test verifies the function with sample input [15, 16, 17, 18] and [w=2](vscode-file://vscode-app/c:/Users/avish/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), ensuring the output includes all valid rolling means [15.5, 16.5, 17.5].